Network Science **8** (S1): S65–S81, 2020
doi:10.1017/nws.2019.64

CAMBRIDGE UNIVERSITY PRESS

ORIGINAL ARTICLE

On fast and scalable recurring link's prediction in evolving multi-graph streams

Shazia Tabassum[1*], Bruno Veloso[2] and João Gama[3]

[1]INESC TEC and FEUP, University of Porto, Porto, Portugal (e-mail: shazia.tabassum@inesctec.pt); [2]INESC TEC, University Portucalense, Porto, Portugal (e-mail: bruno.m.veloso@inesctec.pt) and [3]INESC TEC and FEP, University of Porto, Porto, Portugal (e-mail: jgama@fep.up.pt)
*Corresponding author. Email: shazia.tabassum@inesctec.pt

Action Editor: Hocine Cherifi

Abstract

The link prediction task has found numerous applications in real-world scenarios. However, in most of the cases like interactions, purchases, mobility, etc., links can re-occur again and again across time. As a result, the data being generated is excessively large to handle, associated with the complexity and sparsity of networks. Therefore, we propose a very fast, memory-less, and dynamic sampling-based method for predicting recurring links for a successive future point in time. This method works by biasing the links exponentially based on their time of occurrence, frequency, and stability. To evaluate the efficiency of our method, we carried out rigorous experiments with massive real-world graph streams. Our empirical results show that the proposed method outperforms the state-of-the-art method for recurring links prediction. Additionally, we also empirically analyzed the evolution of links with the perspective of multi-graph topology and their recurrence probability over time.

Keywords: link prediction; evolving network; recurring links; graph streams; temporal networks

1. Introduction

Link prediction is a vital research problem in the area of network science. There are numerous papers addressing this problem from various perspectives and techniques. A summary of it can be found in Al Hasan & Zaki (2011). Most of these methods differ from each other with respect to model complexity, prediction performance, scalability, and generalization ability (Al Hasan & Zaki, 2011). However, new challenges are coming up with the emerging realms of technology, such as the Internet of Things (IoT), Web 3.0, and Smart cities. One of the critical challenges associated is to handle the avalanche of networked data being generated at high speeds. Most of this data flow is a consequence of recurring or repeated links/interactions/communications over time. For example, learning from repeated actions of users in IoT, like to play the favorite songs, make automated breakfast or a reminder to call etc., without the need for a user to manually set the time.

Typically, the link prediction problem is understood in two settings: predict missing links from the same time interval in a static setting and predict new links for a future time step in a temporal/streaming scenario. However, the problem of predicting recurring links has been sparsely studied. Though the models for predicting new links can be expected to work on repeated links as well, but they have been hardly evaluated on this basis (Tylenda *et al.*, 2009). Moreover, the

© Cambridge University Press 2020

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

models for predicting new links tend to exploit indirect relationships or neighborhoods of a node, which is extraneous in the case of recurring links and also impacts scalability.

Stream processing is an exemplary way of processing real-time data. It maintains temporal order which is of utmost importance. Nevertheless, it has its own set of challenges, e.g. one-pass constraints, incremental and decremental characteristics, bounded size algorithms for unbounded data, etc. Therefore, while trying to address these challenges, in this work, we propose to predict recurring links over time with the following contributions:

1. We propose fast and scalable memory-less sample based streaming algorithm for predicting recurring links in evolving weighted/multi-graphs.
2. We empirically demonstrate how the recurring links are related to the past links in a temporal network and their topological significance in it.
3. We carry out extensive experiments with the real world networks as large as 400 million edges. Our empirical results clearly demonstrate the high predictive efficiency achieved by the proposed model in comparison with the state of the art and space efficient techniques.

In an attempt to address the issue of scalability and performance, we implement the method presented by Tabassum & Gama (2018) to provide online samples. Here, we use this methodology to predict recurring links. As the links preserved from this technique are strong, active, and latest, we found from our experiments that they provide a significant accuracy gain. The details of methodology are explained in Section 3.

Rest of the paper is organized as follows: In Section 2, we presented a brief overview of the literature in the area of link prediction. Research works related to the problem of recurring link's prediction are also summarized in Section 2. The proposed model, algorithms, and problem definitions are outlined in Section 3. Section 4 briefs the data networks used and the analysis of recurring links regarding their formation and evolution. Empirical evaluation and a comparative assessment with other baselines are also carried out in Section 4. Finally, the conclusions and future works are summarized in Section 5.

2. Literature review

Link prediction problem was initially more focused on static networks (Liben-Nowell & Kleinberg, 2007). Exploiting a number of node neighborhood-based measures like similarity, proximity and topological features has been a common practice in the literature for predicting new or missing links. Some of these mostly studied features include common neighbors, Jaccard's similarity, Adamic/Adar, preferential attachment, Katz score, hitting time, rooted PageRank, SimRank, shortest distance, clustering index, etc. These features have been used to score and rank the nodes (Liben-Nowell & Kleinberg, 2007) or infused into supervised learning and probabilistic models (Al Hasan *et al.*, 2006; Wang *et al.*, 2007; Benchettara *et al.*, 2010; Wang *et al.*, 2011) to boost their performance. However, the main issue with these models is the number of possible links is quadratic in the number of vertices in a social network, while the number of actual links added to the graph is only a tiny fraction of this number (Al Hasan & Zaki, 2011). Therefore, the methods trying to address this problem are discussed below. As our work mainly deals with the combination of temporal networks, streams, and recurring links, we present below some interesting works in these areas.

2.1 Time aware link prediction methods

Realizing the highly dynamic nature of real-world networks, last decade has been more focused on evolving networks and time aware link prediction methods. A survey of related methods can be found in Marjan *et al.* (2018). However, the basic idea of using proximity measures or structural

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

heuristics (features discussed above) remained the same in most of the works. Papadimitriou *et al.* (2012) found that local neighborhood measures based on 2 hop paths are not as efficient as global network measures. On the other hand, global measures are computationally expensive. In order to provide faster and scalable results, they exploited node neighborhoods of path length l. This gave a better performance but defining l is crucial. However, the method attained the best precision at $l = 3$ for the social networking data sets used. Whilst Sarkar *et al.* (2012) proposed a non-parametric link prediction method considering the local networks of nodes with two hops. Raymond & Kashima (2010) obtained low-rank approximation of similarity matrices for faster link propagation when compared to conjugate gradient-based methods. Song *et al.* (2009) used count-min sketches to approximate path-based proximity measures like Katz, rooted PageRank, and escape probability for large graphs. Their values were incremented for every snapshot. Meanwhile, Rossetti *et al.* (2011) included features from multi-dimensionality of networks. They combined the multi-dimensional versions of node neighborhood features with the temporal information to achieve better performance. Most of the methods listed above tried to provide a faster version in terms of their global counterparts using approximated solutions. Nevertheless, all the methods above have mainly focused on new or missing links and exploiting many computationally extensive features in the case of very large networks.

2.2 Link prediction in streams

With the memory limitations of the conventional systems, it is infeasible to hold massive real-time data in memory and perform multiple passes on it. Streaming data models have gained much interest nowadays due to the current requirements of real-time processing and the nature of unbounded evolving networks. Stream processing solutions have proved to be efficient in presenting plausible models on temporal and evolving data. Some examples include classification (Pan *et al.*, 2014), concept drifts (Gama *et al.*, 2014), triangle counting (Lim *et al.*, 2018), matrix factorization (Matuszyk *et al.*, 2018), change detection (Pereira *et al.*, 2019), estimating global network properties and sampling (Ahmed *et al.*, 2017; Tabassum & Gama, 2016; Zhang *et al.*, 2017), etc.

Though streaming methods were explored in many areas, it is surprisingly less in the area of evolving networks' link prediction. One of the notable works was given by Zhao *et al.* (2016). They followed the suite of adapting the typical heuristics of common neighbors, Jaccard's similarity, Adamic/Adar but in a graph stream setting. To make the approach scalable, they introduced a vertex-biased sampling technique which uniformly sampled the vertices of all degrees without prioritizing high degree vertices. The reported accuracy was almost to the exact method.

2.3 Recurring links prediction

The appearance of new and recurring links in a temporally evolving network creates two distinct types of problems. One is to predict new or missing links, and the other one is to predict recurring links. Considerably, our focus here is on the second one. From the perspective of O'Madadhain *et al.* (2005), networks are described based on two types of relationships: (i) persistent relations, where the relationship between two nodes is stable for a long time. This network is modeled by a simple graph, and new edges are created between vertices that are at a distance two or higher, e.g., friendship, road network, etc.; and (ii) discrete events, where the relationship status is different at every point in time. In this case, the edges may appear not only between vertices at distance two or higher but also between vertices that are already connected (repeating/recurring edges). For example, citation, transaction, and communication networks etc. The authors were interested in the discrete scenario. They trained the classifiers using node neighborhood and event-based features to predict the probabilities of all links in a time series setting. Therefore, the results are mixed and not differential for recurring links.

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

Another interesting work was presented by Tylenda *et al.* (2009). Referring to the above work of O'Madadhain *et al.* (2005), the authors realized two main problems in link prediction: (i) prediction of new links and (ii) prediction of repeated links. They found that the number of repeated links in co-authorship networks is as high as 50% and can reasonably expect similar or more in other networks. Therefore, they predicted repeated links using a time-aware maximum entropy model. The links were forgotten when they violate the constraints imposed by the model, using temporal weights. As their results report, a simple temporal baseline model (last, count) outperformed all the other methods even the authors time-aware maxent. Likewise as in the case of new links. Purposefully, we will discuss in detail about this baseline model in Section 3.4.1.

Zhu *et al.* (2016) adapted an incremental approximate variation of block coordinate gradient descent approach in contrast to the global version, which is computationally prohibitive. The authors evaluated the accuracy of their approach in terms of predicting "all" links (new links and recurring links) and also new links distinctly. Surprisingly, their incremental approach performed weakly than the global counterpart in terms of all links and better in few samples in terms of new links. This shows that the presence of recurring links in "all" links made the predictions worse.

As learned from the previous works, temporal information of the links is very beneficial in predicting links and also natural weights on links, which haven't been explored in any of the works until recently in Moradabadi & Meybodi (2018). Instead, most of the weights assigned to the links are gained from the neighborhoods of the network which are less beneficial in the case of recurring links. Therefore, we propose algorithms by exploiting the above properties of edges.

3. Modeling and predicting recurring links

In this section, we explain the notations, definitions, and the problem statement we are trying to address, followed by our model for predicting links and its complexity analysis. Toward the end of the section, we present other baselines for comparisons.

3.1 Problem definition

We model an evolving network G as a stream of links/edges $\{e_1, e_2, e_3, e_1, e_4 ...\} = E$ which is a multiset. Every edge $e_i = (u, v, t)$ is composed of a pair of vertices from V and a time-stamp t, which indicates the time of occurrence of e_i or simply e which is unique. We assume that the edges are streaming in the order of time stamps. The basic intuition of the proposed approach is for an undirected multi/weighted graph stream. However, a directed graph can be used with a small modification considering an edge between (u, v) different from (v, u).

Definition 1 (Recurring links)**.** *In a multi-graph stream, an edge e is considered recurring if it reoccurs randomly in G at various time stamps t. Another variable τ is a discrete time step/time interval with granularity defined by the user or based on data availability. Initially when $\tau = 0$, we have $G(\tau) = \emptyset$. At every τ, the weight w of an edge e is computed as*

$$w(e, \tau) = \#\ of\ occurrences\ of\ an\ e\ in\ \tau \tag{1}$$

The weights w are incremented in a streaming way sequentially. For weighted graphs, w is the weight of e in the time interval τ.

Definition 2 (Edge vector stream)**.** *Every edge e in G can occur in multiple τ's. Therefore, every edge is a temporal vector $\boldsymbol{a}$ of the state of presence (with a weight w) or its absence in every τ. S is a dynamic sample updating in a streaming fashion. At any time step, S is given as $S(\tau)$. m is the number of unique edges in $S(\tau)$, i.e. $m = |S(\tau)|$. At every τ, $S(\tau)$ contains an instance of $m \times a$ with updated values. $\hat{m}$ is the number of links to be predicted, which can be varied based on the requirement or fixed overtime.*

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

Definition 3 (Problem statement). *Given a temporally evolving network stream until $G(\tau)$, where $\tau = \{0, 1, 2...n\}$, our model aims to predict the probability distribution of edge recurrence in $\tau + 1$.*

3.2 RLP: Recurring link's prediction using temporal bias and frequency

For predicting recurring links with limited space, efficiency and fastness, we propose our model using the dynamic sampling technique (*SBias*) given by Tabassum & Gama (2018). If we consider picking a uniform random sample from a power law distribution (which most of the real-world networks follow), it will likely result in getting all low degree nodes from the long tail. Techniques such as stratified or systematic sampling (Cochran, 2007) would need extra information a priori, like degree distribution of nodes, total population size, variance etc. Moreover, they need multiple passes on data, which are expensive on massive networks. On the contrary, the samples generated using (*SBias*) are proved to preserve certain important properties of temporal networks. They are less biased to disconnected components, low degree nodes, and follow true network distribution (Tabassum & Gama, 2018). Henceforth, we make use of this strategy, and moreover it applies to multi-graphs.

RLP incorporates a forgetting function (RLP_f) from the class of memory-less exponential functions to weight edges as in *SBias*. This function is applied on every unique edge e in $G(\tau)$, after every τ to get $\hat{w}(e, \tau + 1)$, which is the predicted weight of an edge in $G(\tau + 1)$. The function is defined as follows in Equation (2).

$$\hat{w}(e, \tau + 1) = w(e, \tau) + (\hat{w}(e, \tau - 1))(1 - \alpha) \tag{2}$$

The parameter α defines the bias rate and typically lies between 0 and 1 inclusive. In general, this parameter α is chosen in an application-specific way. When α is 1, we forget all of the previous edge occurrences and only maintain information from the latest τ. When $\alpha = 0$, the function returns the global network G with accumulated weights.

We also experimented with another memory less exponential function that is exponential smoothing RLP_{ex} in the place of the above function (2), which is a slight variation of it given as

$$\hat{w}(e, \tau + 1) = \alpha w(e, \tau) + (\hat{w}(e, \tau - 1))(1 - \alpha) \tag{3}$$

However, this change didn't have any effect on the predictions, except increasing slightly the time and space complexities in the case of exponential smoothing variation. For a particular e, the above function with currying can be given as $\hat{w}_e(\tau + 1)$. As this function is applied on every $\mathbf{a}$ in m independently, it explains that for every e, the prediction is not based on its neighborhood or global information. If e does not recur in $\mathbf{a}$, then $\hat{w}_e(\tau + 1)$ is monotonically decreasing. The above function is imposed over m edge vector streams giving m time series. Considering every instance of τ as a vector, we determine the probability distribution of edge recurrences by holding any τ as in Equation (4) for RLP_f and respectively for RLP_{ex}.

$$\hat{w}_{\tau+1}(e) = w_\tau(e) + (1 - \alpha)\hat{w}_{\tau-1}(e) \tag{4}$$

3.2.1 Link labeling

In function (4), for every τ, we observe a power law distribution with a long tail. This long tail if not pruned will be asymptotic to the x-axis and unbounded. The memory-less functions hold the current distribution and discard all the data, but even the current distribution (after every τ) in this case is large enough. Therefore, we prune the outcome at every τ using a threshold θ. The point here worth noting is that in case of *SBias* the threshold is only applied on the part $(\hat{w}_e(\tau - 1))(1 - \alpha)$ of Equation (2), while in *RLP*, we apply the threshold on the total weight, i.e., $\hat{w}_e(\tau + 1)$ at every τ to predict weights in $\tau + 1$. Therefore, the sample generated by *RLP* is less biased to low degree nodes than *SBias*.

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

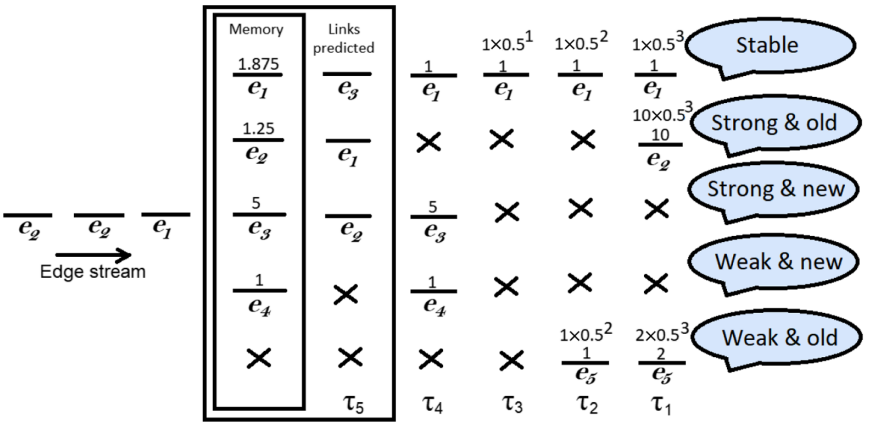


Figure 1. Recurring link prediction model with $\alpha = 0.5$ and $\theta = 1.5$.

The threshold θ can be constant or tuned according to the number of links to be predicted ($\hat{m}$). Here if the updated weight of an edge is greater than or equal to the threshold, then the link is expected to recur. In the Equation below $\hat{y}_e$ is the label of link e at a time step τ.

$$\hat{y}_e = \begin{cases} 1 & \text{if } \hat{w}_\tau(e) \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Where θ lies between 0 and 1 when $\hat{w}_\tau(e)$ is normalized using a constant or min–max technique. If it is not normalized, then θ lies between 0 and the highest $\hat{w}_\tau(e)$ in the network, which would save the extra computation and memory space needed for normalizing.

A higher value of α retains most recent edges in $S(\tau)$ exponentially in the order of τ, while a lower value tends to include older edges also. Likewise, a higher value of θ would retain more strong and stable edges in $S(\tau)$, and a lower value retains less strong and stable ones also. Therefore, we can tune the parameter according to the requirement or the number of links to be predicted ($\hat{m}$). The model is illustrated with a simple example in Figure 1. The weights on the edges are their frequency in the given time step, i.e. w which are not stored. The weights on the edges in the memory are $\hat{w}$. A cross indicates the edge was absent in that τ and the cross in memory that indicate the memory is not used.

3.3 Complexity analysis

After each τ, the edges in the sample with $\hat{w}_\tau(e) \leq \theta$ are deleted from it, retaining stronger and stable edges from m but we still have some weak edges from the latest τ because of the recent bias. Therefore, the sample size $|S(\tau)|$ which is m decreases to $\hat{m}$ at $\tau + 1$ by deleting the edges in long tail at every τ. If e appears again in G at a time step $\tau + i$, it gets added to the $G(\tau)$ again. The space complexity is $O(\hat{m})$ which is the number of edges we need to predict or output size. In the worst case is $O(m)$, which is much less than $|E|$. We use hash maps to store the edges and their weights. So the time complexity is also $O(\hat{m})$ for all the networks or $O(1)$ per edge.

3.4 Baselines with extensions

We evaluated a number of fast and efficient algorithms based on sampling or weighting network.[1] Note that for all the algorithms used, the sample size is not fixed. We varied the sample size/threshold over time τ to match the number of links to predict. Even in the case of reservoir-based samples, the parameter for reservoir size is not treated constant but changing in real time.

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

When the reservoir/sample size changes, the algorithms adapt to the new size. The algorithms are briefly discussed below:

3.4.1 Sort by Last, Count (SLC)

This method was employed by Tylenda *et al.* (2009). It sorts the node pairs by the time of their most recent linkage and the number of links between them. This was proved to be the simplest and most efficient method in the authors' analysis even outperforming their maximum entropy model. We also implemented this in a stream setting. To avoid the sorting complexity, we use the threshold over the distribution as in Section 3.2.1.

3.4.2 Linear Weighting Scheme (LWS)

In this case, all the instances are weighted equally irrespective of their time of existence. We employ this methodology to investigate the significance of the time of occurrence in making predictions. This method treats data as in a static setting. The edges are sorted based on the frequency, and the top edges are considered probable candidates for future links. However, to make the global network scalable enough, we processed it as a stream without considering temporal importance and pruned the links by applying threshold over the weights as in Section 3.2.1.

3.4.3 Space Saving (SS)

Space saving algorithm given by Metwally *et al.* (2005) is a fast and space-efficient alternative for approximately finding frequent items from a data stream. This algorithm maintains the partial interest of information as it monitors only a subset K of items from the stream. It maintains counters for every item in the sample and increments its count when the item re-occurs in the stream. If a new item is encountered in the stream, it is replaced with an item of the least counter value and its count is incremented. This leads to top K items at any time in the stream. As all the algorithms we used above are based on the frequency of link occurrence, we adapted space saving to find most frequent top K links from the edge stream at every τ to predict the recurring links in $\tau + 1$.

3.4.4 Reservoir Sampling for Multi-graphs (RS-M)

Reservoir sampling is a random sampling technique for unbounded data streams proposed by Vitter (1985). This algorithm maintains a reservoir with a predefined sample size m. Firstly the reservoir is filled with the initial items from the stream. Every item after that is computed for the probability m/i of being inserted, where i is the length of the stream exhausted till then. If the probability of the contending item in the stream is greater than the probability of an item in the reservoir which is $1/m$, then uniformly at random an item is picked from the reservoir. The picked item is replaced with the item in the stream. In case the probability is less, the streaming item is discarded. Items in our case are links arriving in the stream.

To do a comparative assessment, we present a multi-graph invariant of the reservoir sampling, while the basic notion remains the same. When a recurring link in the stream has a probability to enter the reservoir, it's weight in the reservoir is incremented by one, without modifying any other links. If a link is deleted from the reservoir, the information about its weight is not stored. When a new link is being added to the reservoir, it is assigned a weight of 1 initially.

3.4.5 Biased Random Sampling for Multi-graphs (BRS-M)

This algorithm is proposed as a biased variant of the Reservoir sampling in Tabassum & Gama (2016). Unlike the above algorithm where the probability of incoming items diminishes as the

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

Table 1. Networks' properties.

Datasets/ Description	#E	#V	#τ	Avg. degree	Avg. weighted degree	Avg. deg per τ	Avg. edge stability	Avg. Recur
Call network	389 994 643	12 213 391	30 (days)	15.570	63.860	0.519	0.072	4.1
DBLP	17 861 201	1 242 068	24 (years)	16.323	28.898	0.223	0.032	1.76
Last.fm band	18 956 166	172 896	51 (months)	10.202	218.780	0.200	0.427	21.27
Radoslaw	82 927	167	237 (days)	69.269	993.140	0.292	0.063	14.34

stream progresses, this algorithm ensures every item goes into the reservoir irrespective of the stream length. An item from the reservoir is chosen for replacement at random. Therefore, the item insertion is deterministic but deletion is probabilistic. An item staying for a long time in the reservoir has a higher probability of getting out than an item inserted recently. Consequently, the items in the reservoir are not biased to very old items as in *RS-M*.

Considering a multi-graph stream, we use the same technique as the above *RS-M* to extend this algorithm. For every recurring link in the reservoir, its weight is updated. For a new link, the assigned weight is 1 and it replaces a link from the reservoir.

3.5 Experimental setup

We perform a set of rigorous experiments over several data sets from slowly evolving networks, like DBLP, to high-speed networks, like phone calls, and as large as billions of links over time. We also choose a bipartite network to recommend items based on recurring links. Below, we first describe the semantics of the data sets followed by an analysis.

3.5.1 Data sets

Here, we present the brief description of data sets used and their properties.

Call network: We use a massive anonymized (Call Detail Records) CDR data provided by a service provider. The average speed of data is 10–280 calls per second around mid-night and mid-day. Calls in the data are associated with time stamps when the call was initiated.

Dblp: This is the collaboration graph of authors of scientific papers from DBLP computer science bibliography which is obtained from KONECT[2] networks. The initial time steps of the available data are too sparse with just one edge per year. Therefore, we considered the 24 years of data toward the end of temporal period. An edge between two authors represents a common publication. Edges are annotated with the date of the publication. There may be multiple edges between two nodes, representing pairs of authors that have written multiple publications together.

Last.fm band: It is a bipartite network of user–band listening events from the music website last.fm.[3] An edge shows that a user listened to a song of a band at a particular time.

Radoslaw: This is an internal email communication network between employees of a mid-sized manufacturing company.[4] The nodes in the network represent employees. Edges between two nodes are individual emails provided with time stamps.

A description of the data sets used appears in Table 1. Remember that we define *Stability of an edge* as the number of time steps it is present in a graph stream over the total number of time steps (Tabassum & Gama, 2018). The average recurrence times of an edge are defined below. The temporal distributions of data are displayed in Figure 2.

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

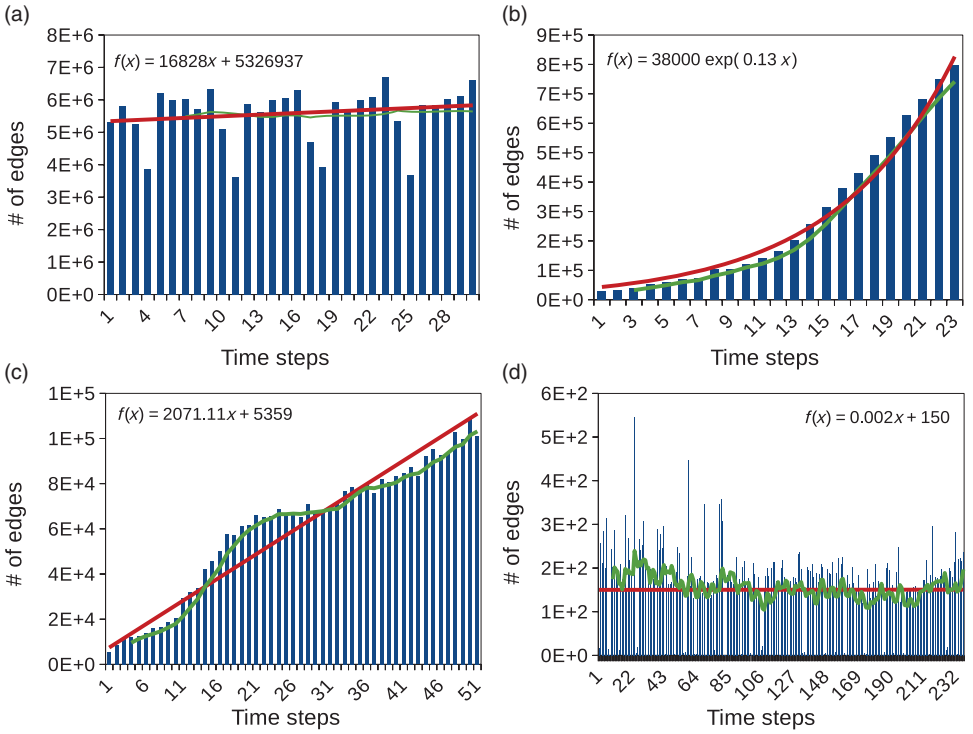


Figure 2. Trend analysis over time. The time series is smoothed using a moving window average (green) and a curve fitted over it to recognize the trend (red). (a) Call network. (b) DBLP. (c) Last.fm band. (d) Radoslaw.

Average recurrences per link (*Avg. recur*)**:** This is a measure to quantify the number of times a link is recurring in a network on an average. We give this by total number of edges over number of unique edges in the data set or average weighted degree by average degree of a network. It can also be referred as the mean of multiplicity of all the edges in the multiset E. This value for different networks is given in Table 1. For a network that do not have recurring links, this value would be equal to 1. The figure shows that on an average, the users in telecom call to a same recipient four times in a month. In DBLP which has the least avg.recur interprets that the authors didn't do the same collaboration more than once or twice (on average) considering directed edges in the data. We can also observe that the users in Last.fm heard on average 21 times the same band in around 4 years. In Radoslaw, the users sent on an average 14 emails to the same person in 237 days. These values indicate the frequency of recurring links in a network.

3.6 Why is it important to predict recurring links?

Tylenda *et al.* (2009) found that 50% of the links in the data they analyzed were repeated links. They expected even more in other data sets. We analyzed the links in our data and recognized their evolution over time by classifying them into two types based on the multi-graph topology: (i) recurring links, i.e., its an edge that occurs multiple times between two known/old nodes and (ii) new links, this can be further subdivided into three categories: i.e., its a new link that occurs between: (a) two old nodes, (b) one old node and one new node, or (c) two new nodes. Most of the link prediction models deal with the first category of new links (between two old nodes) only, while the other two cases are difficult to predict. In Figure 3, we indicate the percentage of recurring or new links at any time step τ corresponding to the links in 0 to $\tau - 1$. One can see that the recurring links in the figure are making the highest percentage and are gradually increasing.

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

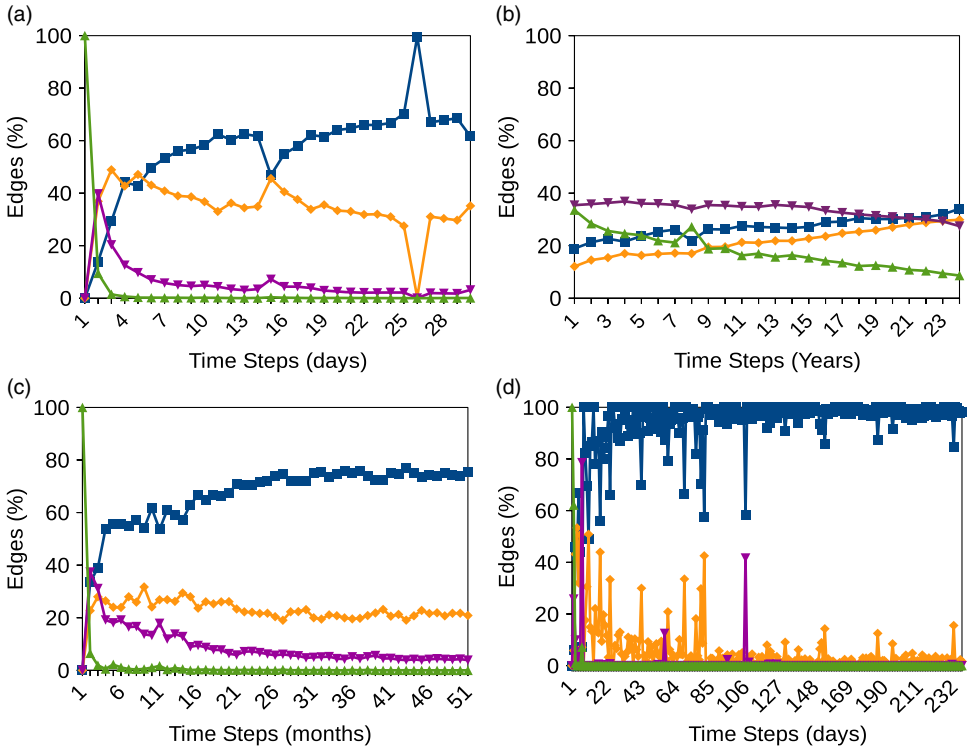


Figure 3. Link analysis over time with the multi-graph perspective. (a) Call network. (b) DBLP. (c) Last.fm band. (d) Radoslaw.
■ Repeated links between two old nodes (Recurring)
◆ New links between two old nodes (New)
▼ New links between one old and one new node (New)
▲ New links between two new nodes (New).

In some instances, 100% of the future links have reoccurred. The analysis reflects the apparent behavior of users in the networks. In the case of Call network and Radoslaw data, the users are more interested in communicating with the old pals than making new contacts. So as in DBLP if two authors publish a paper, most probably they will publish a second paper, but in DBLP one can observe that the new connections are also much appreciated than in the other three networks. This may lead to the exponential growth of network size which was shown in Figure 2. Last.fm has a similar trend where the users are interested in listening to the same bands as earlier, and fewer users try new bands. Therefore, it is highly significant to predict recurring links as they make a large fraction of future links in most of the real-world networks.

3.7 How are the reoccurred links associated with past links?

Chen *et al.* (2013) found the young links to be more informative than the mid-age links and much more informative than old links in a temporal network, while their influential power differs with data. We analyzed our data sets to check what proportion of the links in the last time step τ have already occurred in the previous time steps $\tau - 1, \tau - 2, \ldots$. To derive a space efficient model, we kept on deleting the edges already found. Figure 4 shows the re-occurrence probability of the links in the last time step based on previous time steps for different networks. Distance on x-axis indicates the distance between the current τ and previous $\tau's$, i.e., the distance between τ and

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

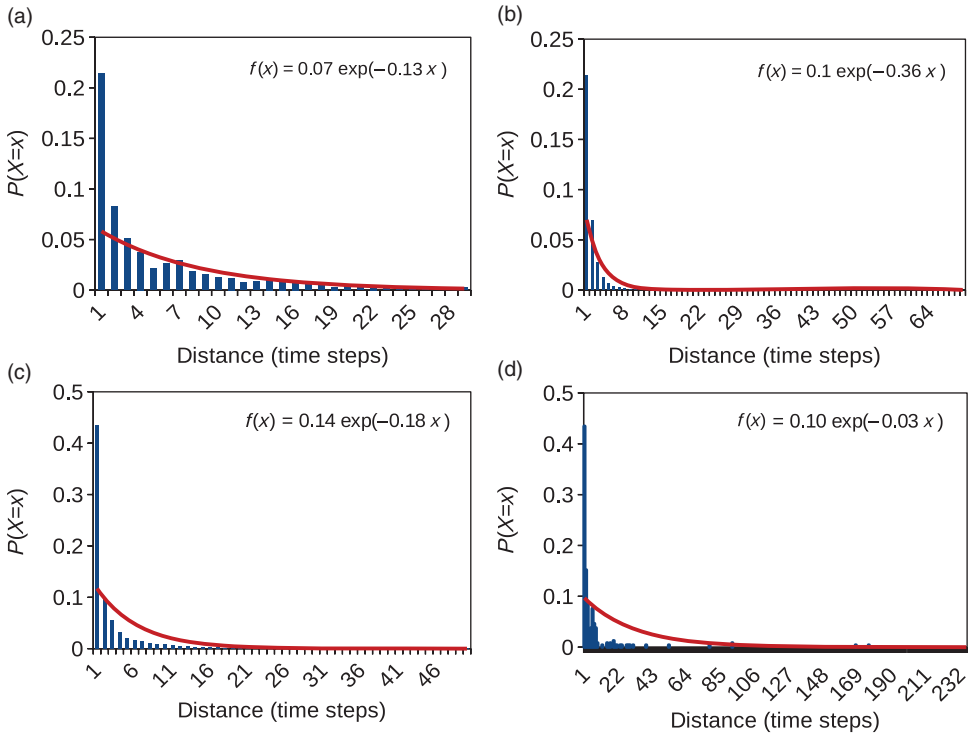


Figure 4. Recurrence probability of links. The bars (blue) indicate the empirical mass probability of links from τ that has recurred from the previous time steps at a distance given on x-axis. The curve (red) is a fitted function over it. (a) Call network. (b) DBLP. (c) Last.fm band. (d) Radoslaw.

$\tau - 1$ is 1. One can observe that most recent links to τ have the highest probability of recurrence in it, and the probability is decreasing exponentially as the temporal distance to τ is increasing. We used maximum likelihood estimation to find the value of α in the exponential function which differs with data sets.

4. Experimental evaluation

4.1 Prequential evaluation for time series graphs

To evaluate and compare the given temporal models, we followed prequential or predictive sequential evaluation methodology (Gama *et al.*, 2013; Oliveira *et al.*, 2018). It is one of the standard frameworks for evaluating models in a stream setting. The initial sequence of examples is learned to predict the upcoming examples in the stream. Then, an error is calculated using a loss function ($L(y_i, \hat{y}_i)$) between the true values and the predicted values. For every next iteration, the true values are combined with the learned sequence to predict the next upcoming values. The number of examples to be learned can be growing or forgotten using weights. In this work, we implement it over temporal network data. The sequence of items is a time series, with every time step having a subset of all the links in the network. As given in Equation (6), the loss function is calculated for all the $\hat{m}$ number of links predicted at every τ. In the case of *LWS*, we learn the time series from 0 to $\tau - 1$ to predict τ and 0 to τ to predict $\tau + 1$. For *RLP*, the predicted links from the previous time step and the true links of that time step are used to predict the links in

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

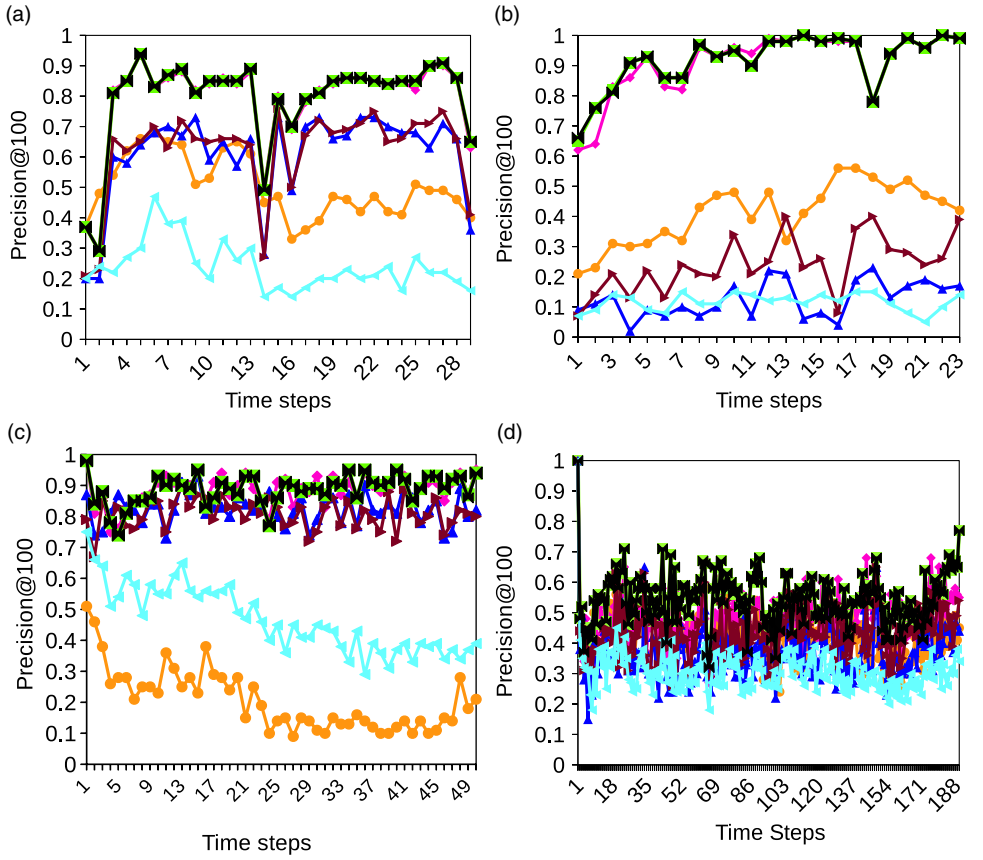


Figure 5. Precision@K over time. (a) Call network. (b) DBLP. (c) Last.fm band. (d) Radoslaw.
RLP_f RLP_{ex} *SLC* *LWS* *SS* *BRS-M* *RS-M*.

the next time step. *SLC* uses only the true links from the previous time step. For all the other algorithms, the updated samples for every time step are used for making predictions. Therefore, we exhibit the results for every time step in Figure 5. The loss function used in this plot is precision@k. In Figure 6, we calculated the area under precision-recall curve by computing the loss functions (precision and recall) multiple times for each time-step by varying thresholds. More detailed explanations for plots are given in the sections below. To facilitate the comprehensibility of experiments, we listed the abbreviations of models in Table 2

$$\epsilon(\tau) = L(\hat{y}_e(\tau), y_e(\tau)); \text{ where } e = 1, 2, ...\hat{m} \tag{6}$$

4.2 Temporal evaluation of positive predictive value

To evaluate the preciseness of models over time, we fixed the threshold such that the number of links to be predicted at every τ is constant K. Precision@K is a useful metric where they can be millions of predicted items but not all of them are necessary, for example, in the case of search queries, recommendations etc. Figure 5 shows the precision@100 over time in different data sets using all the models. To make the K value consistent across all the data sets per τ, we choose it not too small for the largest data set and not too high for the smallest data we have. For all the

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

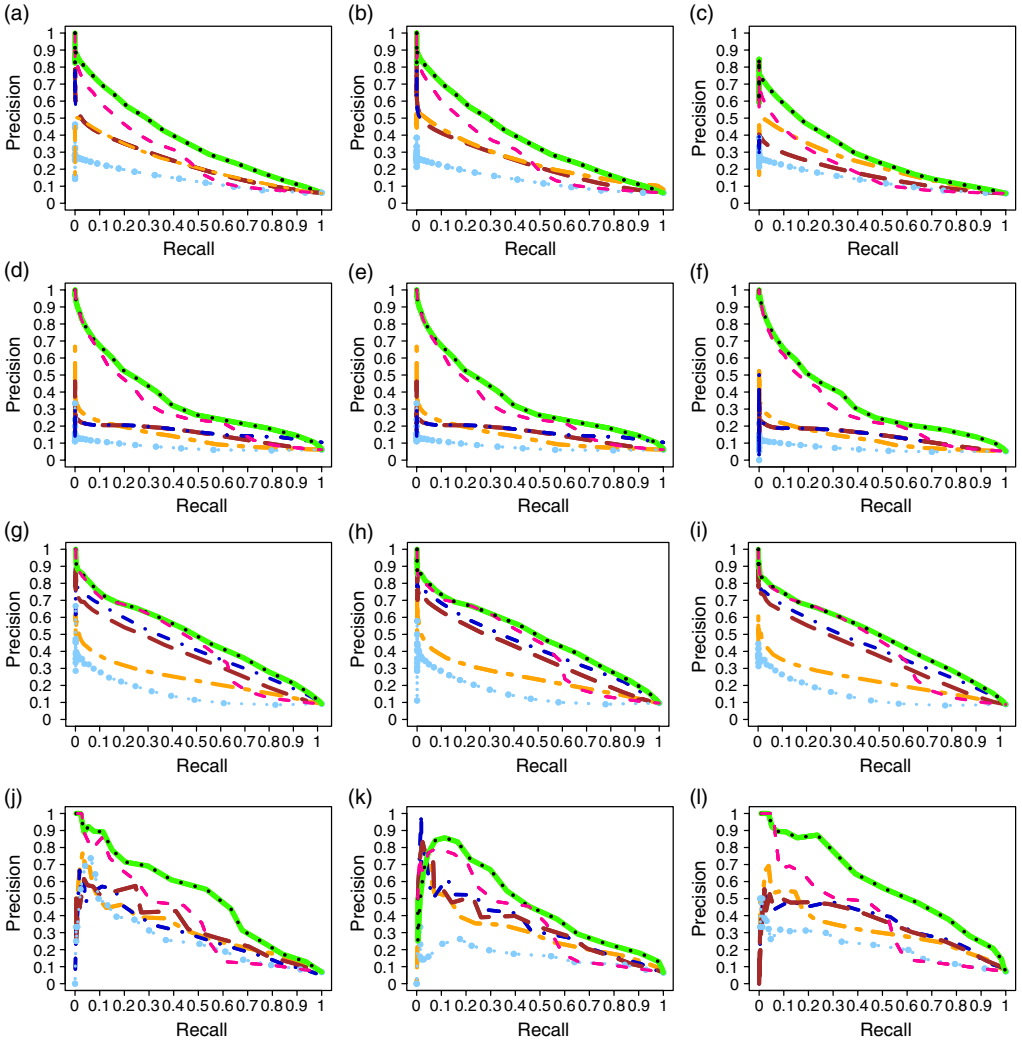


Figure 6. PR-curves with varying thresholds for different models. (a) Call network($\tau - 2$). (b) Call network($\tau - 1$). (c) Call network(τ). (d) DBLP($\tau - 2$). (e) DBLP($\tau - 1$). (f) DBLP(τ). (g) Last.fm band($\tau - 2$). (h) Last.fm band($\tau - 1$). (i) last.fm band(τ). (j) Radoslaw($\tau - 2$). (k) Radoslaw($\tau - 1$). (l) Radoslaw(τ).
(▪ ▪ ▪ ▪) *RLP*$_f$, (——) *RLP*$_{ex}$, (– – –) *SLC*, (—·—) *LWS*, (—·—·—) *SS*, (——) *BRS-M*, (·····) *RS-M*.

networks except DBLP, we see a low variance in precision over time. This difference is due to the network evolution trend in DBLP as can be seen in Figure 2. The growth of edges in DBLP is exponential over time, while in other networks is linear or constant. We observe poor results with the Radoslaw data (Figure 2d) because of lot of missing data. We have one or two time steps missing every week in the temporal order of data where it shows dips in the performance. However, it is evident that RLP_f and RLP_{ex} exhibit the best average performance over time.

4.3 Prediction efficiency using PR-Curves

Precision-Recall curves evaluate the percentage of true positives among positive predictions with different threshold values. They are known to be more informative than the receiver operating

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

Table 2. Recurring link prediction models.

Algorithm	Model abbreviation
Recurring link prediction with forgetting	RLP_f
Recurring link prediction with exponential smoothing	RLP_{ex}
Sorting based on last, count	*SLC*
Linear weighting scheme	*LWS*
Space saving	*SS*
Reservoir sampling for multi-graphs	*RS-M*
Biased random sampling for multi-graphs	*BRS-M*

characteristic curves for evaluating future classification performance in binary class imbalanced problems. They can be more reliable in this case of sparse data sets with very few links between pairs of nodes and heavily right-skewed distributions. We present in Figure 6 the PR curves for the predictions of the last three time steps. These results are achieved by performing multiple iterations with varying thresholds per τ. But for each iteration, the threshold is maintained across all models. For RLP_f and RLP_{ex}, we present the curves for α with highest AUC-PR. Though the area under the curve for all the other values of α is still higher than that of other algorithms. From the figure, it is evident that *RLP* clearly outperforms in all the data sets and time steps.

4.4 Running time evaluation

In Figure 7, we report the running cost of all the algorithms. To plot the given figure, all the networks were fully processed in the streaming way till the last time step while changing the sample size (m) stored by the algorithms in memory, which is given on x-axis. The experiments were performed on intel core an i7@3.40 GHz processor with 32 GB memory. From the figure, it is evident that space saving exhibits a very slow performance in time and increasing exponentially with sample size. All the other algorithms show almost constant time performance even with the increase in size. Reservoir and random sampling are simple and fast techniques and show the best performance up to a point for comparatively small data sets. As for the very large sample sizes the slope of reservoir and random sampling curves increases, while RLP_f performs consistently. While RLP_f and RLP_{ex} gives equal results for ranking predictions in the above experiments, RLP_{ex} is a bit lagging in time efficiency.

4.5 Discussion

In this section, we summarized the results and presented possible explanations. We empirically show that our model using *RLP* gives better predictions than other baseline models referred in all the data sets. It performs well with an average precision as good as 100% in the top 100 predictions as shown in Figure 5. *RLP* gives the best predictions with the lastfm.band bipartite data which indicates that it can be beneficial to improve the performance of recommendation models.

One of the crucial questions in the above experiments is the estimation of the value of forgetting factor α. We have observed that for all the values of α between 0 and 1, the results still outperform given baselines. For brevity, we could not display all. However we estimated the value of α using MLE in Section 3.7 on recurrence probability given in Figure 4. The exponential curve is fit from $\tau - 1$ as RLP_f applies α from $\tau - 2$ to predict links in τ (Equation (2)), while $\tau - 1$ is given weight 1. From our experiments, over time we have found that higher efficiency is attained around $\alpha \pm 0.1$ over all the time steps, but unique to a data set. $\alpha \pm 0.1$ indicated in Figure 4 gave the best predictions in the case of respective data sets.

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

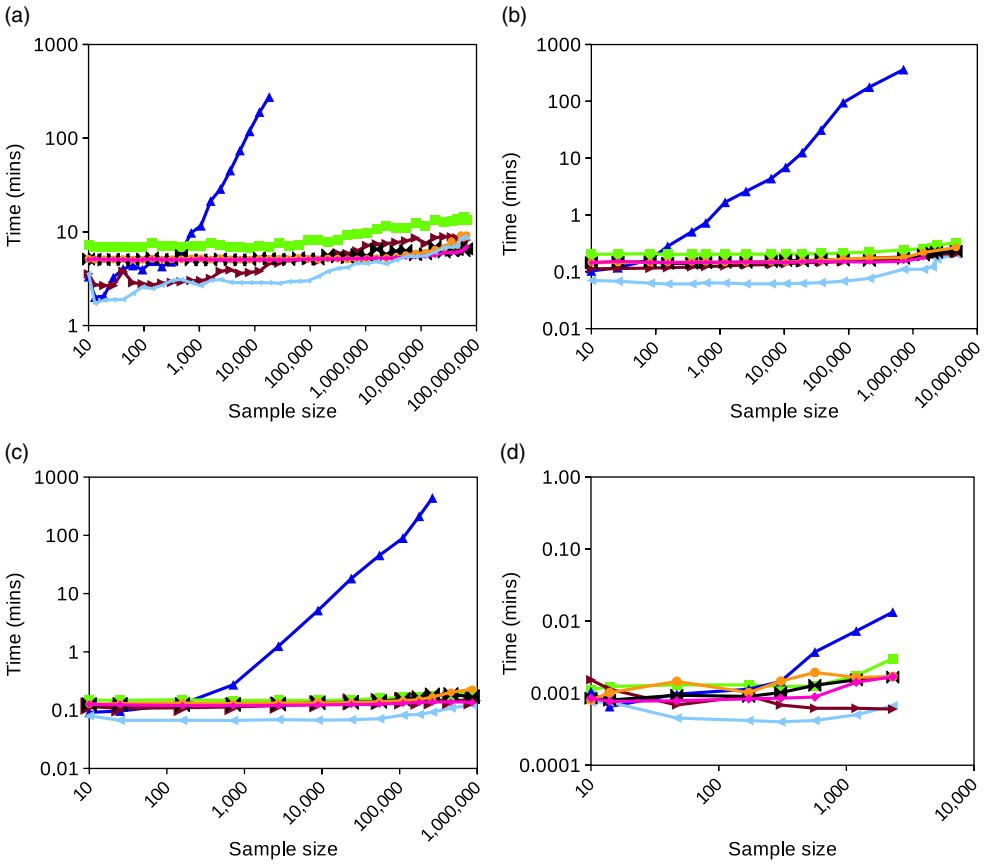


Figure 7. Running time by increasing output size $\hat{m}$ while the entire network over all time-steps is processed in a streaming way. (a) Call network. (b) DBLP. (c) Last.fm band. (d) Radoslaw.
⋈ *RLP*$_f$ ■ *RLP*$_{ex}$ ◆ *SLC* ● *LWS* ▲ *SS* ▶ *BRS-M* ◀ *RS-M*.

We also notice that the simple random sampling model (*BRS-M*) performs better or as good as the higher time complexity space saving algorithm in finding top links (Figures 6 and 5). It is also apparent that reservoir sampling (*RS-M*) performs very poorly in most of the cases.

Independent of the main results in this paper, we also found that exploiting temporal information can be particularly beneficial for recurrent link prediction, which is in accordance with previous studies of Tylenda *et al.* (2009). As *LWS* which is a static model performs weaker than other models exploiting temporal information.

5. Conclusions and future works

Recurring links make a significant fraction of network links and keep on increasing, which is also empirically demonstrated in this work. We have also observed the probability of a link recurrence decreases exponentially as a function of time. Therefore in this paper, we presented simple, space- and time-efficient heuristics to predict recurring links based on the past data, without exploiting the time and space expensive neighborhood or global information of the network for each node/link. We proposed *RLP* which weights the links exponentially based on their frequency and recency in the stream. It is fast and scalable with $O(1)$ per edge and almost constant running time with the increase in sample size as seen empirically. We performed an extensive experimental

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

comparison of the proposed method against existing link prediction algorithms, using massive real-world data sets from different domains. *RLP* exhibits high precision and recall over previously proven efficient method for recurring links prediction. We also extended and incorporated other state-of-the-art algorithms such as space saving, reservoir sampling, and random stream sampling algorithms for link prediction.

For future works, we would like to extend our model for new links' prediction. The analysis of a variety of networks with the perspective of recurrent links has not been carried out so far. We would try to advance the analysis of networks presented in the paper to understand and model the growth of networks.

Acknowledgments. This work is financed by National Funds through the Portuguese funding agency, FCT—Fundação para a Ciência e a Tecnologia within project: UID/EEA/50014/2019. The authors also thank WeDo Business for providing the anonymized CDR data set.

Conflict of interest. None

Notes

1 Java code for all the algorithms is available at https://github.com/ShaziaTabassum/Evolving-Multigraph-Stream/tree/master/networks/src/dynamic/sampling.
2 Data available at http://konect.uni-koblenz.de/networks/dblp_coauthor.
3 Data available at http://konect.uni-koblenz.de/networks/lastfm_band.
4 Data available at http://konect.uni-koblenz.de/networks/radoslaw_email.

References

Ahmed, N. K., Duffield, N., Willke, T. L., & Rossi, R. A. (2017). On sampling from massive graph streams. *Proceedings of the VLDB Endowment*, *10*(11), 1430–1441.

Al Hasan, M., Chaoji, V., Salem, S., & Zaki, M. (2006). Link prediction using supervised learning. In *Sdm06: Workshop on link analysis, counter-terrorism and security*.

Al Hasan, M., & Zaki, M. J. (2011). A survey of link prediction in social networks. In Charu C. Aggarwal (Ed.), *Social network data analytics* (pp. 243–275). Boston, MA: Springer.

Benchettara, N., Kanawati, R., & Rouveirol, C. (2010). Supervised machine learning applied to link prediction in bipartite social networks. In *2010 international conference on advances in social networks analysis and mining* (pp. 326–330). IEEE.

Chen, H.-H., Miller, D. J., & Giles, C. L. (2013). The predictive value of young and old links in a social network. In *Proceedings of the ACM SIGMOD workshop on databases and social networks* (pp. 43–48). DBSocial'13. New York, NY, USA: ACM.

Cochran, W. G. (2007). *Sampling techniques*. Hoboken, NJ: John Wiley & Sons.

Gama, J., Sebastião, R., & Rodrigues, P. P. (2013). On evaluating stream learning algorithms. *Machine Learning*, *90*(3), 317–346.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, *46*(4), 44.

Liben-Nowell, D., & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, *58*(7), 1019–1031.

Lim, Y., Jung, M., & Kang, U. (2018). Memory-efficient and accurate sampling for counting local triangles in graph streams: From simple to multigraphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *12*(1), 4.

Marjan, M., Zaki, N., & Mohamed, E. A. (2018). Link prediction in dynamic social networks: A literature review. *2018 IEEE 5th international congress on information science and technology (CIST)* (200–207). IEEE.

Matuszyk, P., Vinagre, J., Spiliopoulou, M., Jorge, A. M., & Gama, J. (2018). Forgetting techniques for stream-based matrix factorization in recommender systems. *Knowledge and Information Systems*, *55*(2), 275–304.

Metwally, A., Agrawal, D., & El Abbadi, A. (2005). Efficient computation of frequent and top-k elements in data streams. In *International conference on database theory* (pp. 398–412). Springer.

Moradabadi, B., & Meybodi, M. R. (2018). Link prediction in weighted social networks using learning automata. *Engineering Applications of Artificial Intelligence*, *70*, 16–24.

Oliveira, M., Torgo, L., & Costa, V. S. (2018). Evaluation procedures for forecasting with spatio-temporal data. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 703–718). Springer.

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press

O'Madadhain, J., Hutchins, J., & Smyth, P. (2005). Prediction and ranking algorithms for event-based network data. *ACM SIGKDD Explorations Newsletter*, *7*(2), 23–30.

Pan, S., Wu, J., Zhu, X., & Zhang, C. (2014). Graph ensemble boosting for imbalanced noisy graph stream classification. *IEEE Transactions on Cybernetics*, *45*(5), 954–968.

Papadimitriou, A., Symeonidis, P., & Manolopoulos, Y. (2012). Fast and accurate link prediction in social networking systems. *Journal of Systems and Software*, *85*(9), 2119–2132.

Pereira, F. S. F., Tabassum, S., Gama, J., de Amo, S., & Oliveira, G. M. B. (2019). Processing evolving social networks for change detection based on centrality measures. In M. Sayed-Mouchaweh (Ed.), *Learning from data streams in evolving environments* (pp. 155–176). Switzerland: Springer.

Raymond, R., & Kashima, H. (2010). Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 131–147). Springer.

Rossetti, G., Berlingerio, M., & Giannotti, F. (2011). Scalable link prediction on multidimensional networks. *2011 IEEE 11th international conference on data mining workshops* (pp. 979–986). IEEE.

Sarkar, P., Chakrabarti, D., & Jordan, M. (2012). Nonparametric link prediction in dynamic networks. arxiv preprint arxiv:1206.6394.

Song, H. H., Cho, T. W., Dave, V., Zhang, Y., & Qiu, L. (2009). Scalable proximity estimation and link prediction in online social networks. In *Proceedings of the 9th ACM SIGCOMM conference on internet measurement* (pp. 322–335). ACM.

Tabassum, S., & Gama, J. (2016). Sampling massive streaming call graphs. In *Proceedings of the 31st annual ACM symposium on applied computing* (pp. 923–928). ACM.

Tabassum, S., & Gama, J. (2018). Biased dynamic sampling for temporal network streams. In *International conference on complex networks and their applications* (pp. 512–523). Springer.

Tylenda, T., Angelova, R., & Bedathur, S. (2009). Towards time-aware link prediction in evolving social networks. In *Proceedings of the 3rd workshop on social network mining and analysis*. SNA-KDD'09 (pp. 9:1–9:10). New York, NY, USA: ACM.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, *11*(1), 37–57.

Wang, C., Satuluri, V., & Parthasarathy, S. (2007). Local probabilistic models for link prediction. In *Seventh IEEE international conference on data mining (ICDM 2007)* (pp. 322–331). IEEE.

Wang, D., Pedreschi, D., Song, C., Giannotti, F., & Barabasi, A.-L. (2011). Human mobility, social ties, and link prediction. In *Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1100–1108). ACM.

Zhang, J., Zhu, K., Pei, Y., Fletcher, G., & Pechenizkiy, M. (2017). Clustering-structure representative sampling from graph streams. In *International workshop on complex networks and their applications* (pp. 265–277). Springer.

Zhao, P., Aggarwal, C., & He, G. (2016). Link prediction in graph streams. In *2016 IEEE 32nd international conference on data engineering (ICDE)* (pp. 553–564). IEEE.

Zhu, L., Guo, D., Yin, J., Ver Steeg, G., & Galstyan, A. (2016). Scalable temporal latent space inference for link prediction in dynamic social networks. *IEEE Transactions on Knowledge and Data Engineering*, *28*(10), 2765–2777.

Cite this article: Tabassum S., Veloso B., and Gama J. (2020). On fast and scalable recurring link's prediction in evolving multi-graph streams. *Network Science* **8**, S65–S81. https://doi.org/10.1017/nws.2019.64

https://doi.org/10.1017/nws.2019.64 Published online by Cambridge University Press